Mephisto: an Open Source WIFI OSC Controller for Faust Applications

Romain MICHON
Center for Computer Research in Music and Acoustics
Department of Music,
Stanford, CA 94305-8180,
USA,
rmichon@ccrma.stanford.edu

Abstract

MEPHISTO is a small battery powered open source Arduino based device. Up to five sensors can be connected to it using simple 1/8" stereo audio jacks. The output of each sensor is digitized and converted to OSC messages that can be streamed on a WIFI network to control the parameters of any FAUST generated app.

Keywords

Faust, Arduino, Controller, OSC

1 Introduction

In the past few years, the FAUST[1] [Orlarey and Letz, 2002] programming language has been used increasingly by researchers and developers to implement new algorithms for real time audio signal processing. As a result, dozens of open source FAUST effects and synthesizers are now freely available. For example, Julius Smith's libraries [Smith, 2012] and the FAUST-STK [Michon and Smith, 2011] provide a large array of objects ranging from the simplest lowpass filter to complex feedback delay networks and physical models of musical instruments.

However, we observed that these technologies remain relatively inaccessible to musicians who don't have the knowledge (and the desire) to compile a FAUST object on their laptop. In other words, these elements are not "plug and play".

One of the tool already at our disposal to facilitate the sharing and the use of FAUST objects is the Online Compiler[2] [Michon and Orlarey, 2012]. This web app contains an interactive catalog of FAUST programs that can be compiled to any of the available FAUST architectures and then downloaded. Users can easily add their own FAUST codes to the catalog or modify existing elements. Even if this very high level tool makes the creation of plug-ins, etc., very easy, it targets users who have some knowledge in computer music and who know how to use a VST[3], an external audio interface, etc.

Thus, to make things even easier we started to think about a FAUST stomp box that could be based on an embedded Linux system such as a *Raspberry Pi*[4]. It would have been able to connect to the online compiler to provide its user a list of the objects stored in the catalog. A download button to cross compile and then download the effect or the synthesizer in the FAUST box would have made the use of this system very easy.

However, even though *Raspberry Pis* are great prototyping platforms, their computation power is quite limited. Also, their booting time can be a problem for impatient musicians. We then realized that a smartphone or a tablet could do a similar job and would be more user friendly. While there already existed a `faust2ios` architecture, Apples product were presenting a huge disadvantage over Android phones: in order to be installed, an app has to be approved by the *Apple Store* which was making our concept of a customizable stomp box impossible to implement. Thus we opted for Android and created a `faust2android` [Michon, 2013] architecture.

Another key component of our project was to provide an easy way for musicians to control the different parameters of a FAUST object during a live performance. While smartphones offer built-in basic controllers: touch screen, accelerometer, etc., these elements are not very practical to interact with if the user is playing an instrument and processing its sound with his phone. Indeed, many instruments require the use of both hands, making it inconvenient to interact with another interface.

MEPHISTO was created to solve this problem.

[1] `http://faust.grame.fr`.
[2] `http://faust.grame.fr/compiler`.
[3] *Virtual Studio Technology*.
[4] `http://www.raspberrypi.org/`.

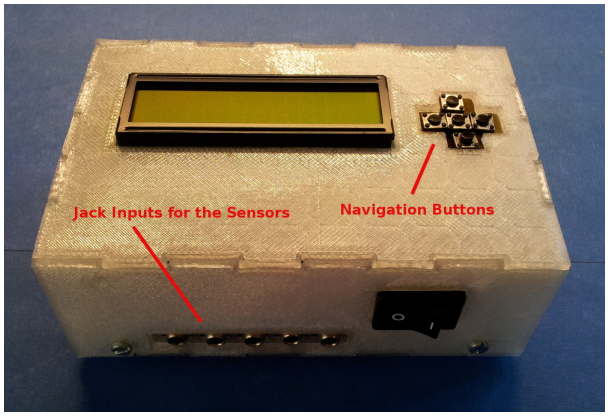

Figure 1: View of MEPHISTO from its top.

It is a small battery powered device that can be easily attached on someone's belt (cf., figures 1 and 2). Up to five sensors can be connected to it using simple 1/8" stereo audio jack plugs. The output of each sensor is digitized and converted to OSC[5] [Wright, 2005] messages that can be streamed on a WIFI network to control the parameters of any FAUST generated app. As OSC is a standard protocol, MEPHISTO can be used with `faust2android` apps but is also compatible with most of the FAUST architectures and programs enabling OSC communication.

As a "DIY"[6] open source project, MEPHISTO only uses open source hardware (Arduino, etc.) and was designed to be easily built by anyone. A web page giving the instructions to build your own MEPHISTO has been created[7].

2 Hardware

Designing small scale open source hardware can be a rather complicated task. Indeed, while software can be easily deployed and shared, in many cases hardware requires a production chain, etc. For this reason, MEPHISTO has been designed to be easily and quickly built by anyone.

2.1 The Case

To make it as easy as possible for users, the case of MEPHISTO is 3D printable. It has been designed with Blender[8] which is an open source program for 3D design that has some CAD features.

[5] *Open Sound Control*: `http://opensoundcontrol.org/`.

[6] Do It Yourself.

[7] `http://ccrma.stanford.edu/~rmichon/mephisto`.

[8] `http://www.blender.org/`.

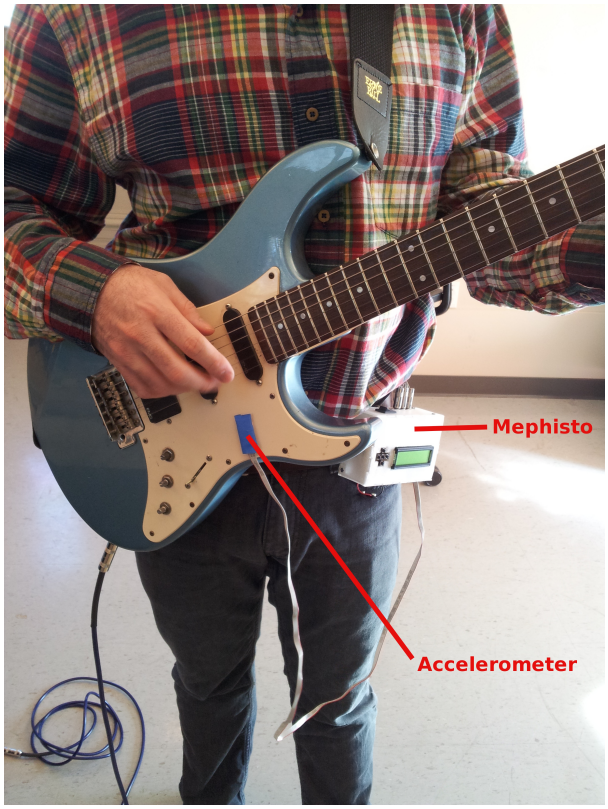

Figure 2: Use example of MEPHISTO.

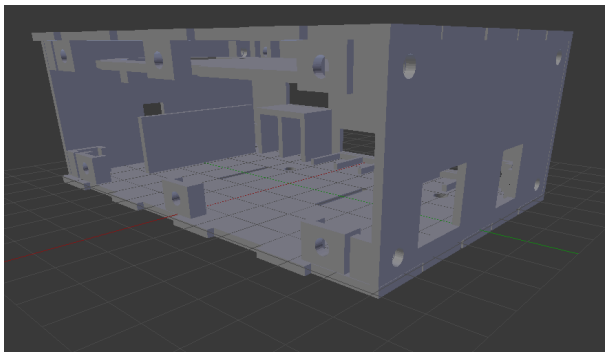

Figure 3: 3D model of MEPHISTO's case as it appears in `Blender`.

We're perfectly aware that not everyone has a 3D printer at home, but 3D printed models can now be ordered very easily online for very cheap. Moreover, MEPHISTO's case has a very simple design and can be printed on almost every 3D printers.

2.2 Electronics

MEPHISTO is based on an Arduino Uno[9] and a WIFI Shield[10] (cf., figure 4). The Arduino provides five analog inputs that are used in

[9] `http://arduino.cc/en/Main/arduinoBoardUno`.

[10] `http://arduino.cc/en/Main/ArduinoWiFiShield`.

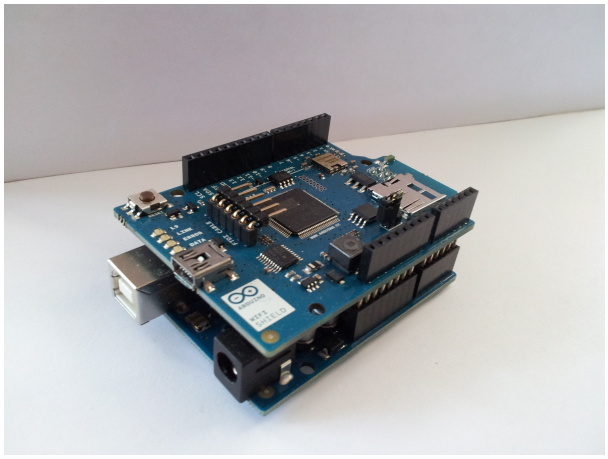

Figure 4: An Arduino Uno and its WIFI shield.

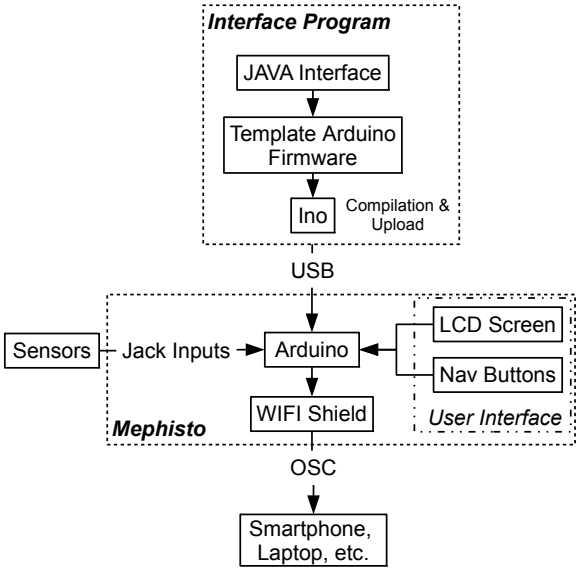

Figure 5: MEPHISTO flow chart.

MEPHISTO to digitize the output signals of the sensors. Simple 1/8" stereo audio jacks (three pins) are used to bring power to the sensors and to retrieve their output signal (cf., figure 1).

Users can configure basic parameters such as the WIFI network to connect to or the IP address of the host using an LCD screen and a navigation button interface.

MEPHISTO can be powered with any DC power adapter between seven and nine volts or with a simple nine volts battery. We considered using lithium ion batteries instead but these are more expensive and need a special charger. With five simple sensors plugged to it, MEPHISTO can run for about four hours on the same nine volts battery. Moreover, it is very easy and quick to replace it.

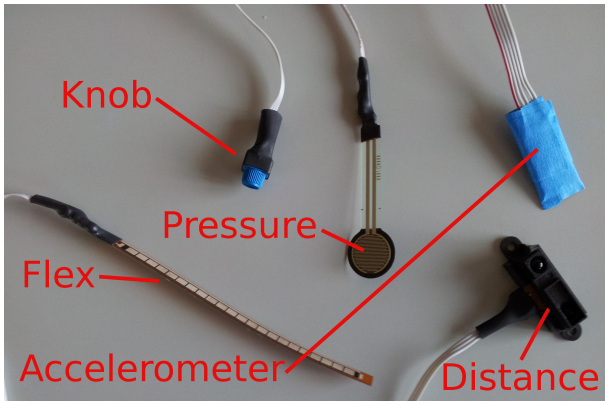

Figure 6: Example of sensors that can be connected to MEPHISTO with their 1/8" jack plugs.

Dozens of sensors have been tested and can be easily prepared to work with MEPHISTO. Our website explains how to set up an accelerometer, a pressure sensitive and a flex sensors, trim pots, etc.[11].

3 Software

3.1 Arduino Firmware

The Arduino firmware carries out a large number of tasks. It retrieves and digitizes the output signals of the sensors and scale them in function of the parameters specified in the interface program (cf., §3.2). Then it converts them to OSC messages using `oscuino`[12]. The OSC address of each sensor can be configured in the interface program (cf., §3.2).

The firmware also handles the user interface implemented through the LCD screen and the navigation buttons.

3.2 Interface JAVA Program

Even though MEPHISTO provides its own very simple interface to configure it by the mean of its LCD screen and navigation buttons, a `JAVA` program[13] that can run on both Linux and MacOSX was created to carry out this task more precisely.

This simple program allows to configure the OSC address and the range of the OSC messages sent by MEPHISTO for each jack input. It is also possible to choose which sensor is connected to which jack in order to carry out some scaling on their output signal (even if the MEPHISTO website explains how to do basic electronic scaling

[11] `http://ccrma.stanford.edu/~rmichon/mephisto`.
[12] `http://cnmat.berkeley.edu/oscuino`.
[13] `https://github.com/rmichon/mephisto/`.

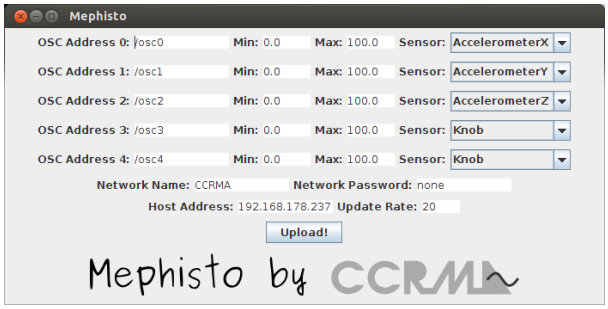

Figure 7: Screenshot of the interface program used to configure MEPHISTO from a desktop.

on sensors, it is often necessary to adjust their output range computationally).

The MEPHISTO configuration program also makes it possible to pre-configure the WIFI network to which mephisto will connect as well as its password if it is protected, the IP address of the host and the rate at which the messages are sent.

This interface program formats and customizes the Arduino firmware in function of the provided parameters. It then compiles it and uploads it to the Arduino if it is connected to one of the USB port using `ino`[14]. As this program only works with Linux and MaxOSX it makes the interface only usable on these platforms even though it can also be executed on Windows.

4 Conclusion

MEPHISTO is an open source project that improves and simplifies the control of sound effects and synthesizers running on a mobile device. Any kind of sensor can be connected to it and used as on OSC controller for live performance.

The FAUST online compiler together with MEPHISTO, `faust2android` and the FAUST catalog of sound effects and synthesizers greatly simplifies the use of FAUST objects by musicians.

The use of 3D printing in the framework of open source hardware projects makes things a lot easier than in the past. Indeed users don't need to have any background in manufacturing and only have to take care of putting the different components together.

Similarly, Arduinos are relatively self contained environments that significantly reduce the size of electronic circuits making projects like MEPHISTO easy to build at home.

[14] `http://inotool.org/`.

References

Michon and Orlarey. 2012. The faust online compiler: a web-based ide for the faust programming language. In *Proceedings of the Linux Audio Conference (LAC-2012)*, pages 111–116, Stanford University, USA.

Michon and Smith. 2011. Faust-stk: a set of linear and nonlinear physical models for the faust programming language. In *Proceedings of the Conference on Digital Audio Effects (DAFx-11)*, pages 199–204, IRCAM, Paris, France.

R. Michon. 2013. Faust2android: a faust architecture for android. In *Proceedings of 16th Int. Conference on Digital Audio Effects (DAFx-13)*, pages 301–304, National University of Ireland, Maynooth, Ireland.

Fober Orlarey and Letz. 2002. An algebra for block diagram languages. In *Proceedings of the International Computer Music Conference (ICMA)*, pages 542–547, Gothenburg, Sweden.

J. Smith. 2012. Signal processing libraries for faust. In *Proceedings of the Linux Audio Conference (LAC-2012)*, pages 33–38, Stanford University, USA.

M. Wright. 2005. Open sound control: an enabling technology for musical networking. *Organised Sound*, 10(03):193–200.